UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 10/802,586 | 03/17/2004 | Joel David Munter | MP1502 | 5149 |

64768 7590 04/15/2009

MARSHALL, GERSTEIN & BORUN, LLP (MARVELL)
233 SOUTH WACKER DRIVE
6300 SEARS TOWER
CHICAGO, IL 60606-6357

| EXAMINER | |
|---|---|
| WANG, BEN C | |

| ART UNIT | PAPER NUMBER |
|---|---|
| 2192 | |

| MAIL DATE | DELIVERY MODE |
|---|---|
| 04/15/2009 | PAPER |

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

| Office Action Summary | Application No. | Applicant(s) | |
|---|---|---|---|
| | 10/802,586 | MUNTER ET AL. | |
| | Examiner | Art Unit | |
| | BEN C. WANG | 2192 | |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --*

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on *18 December 2008*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) *1-11,13-18,20-26 and 28-33* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-11, 13-18, 20-26, and 28-33* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

## DETAILED ACTION

1. Applicant's amendment dated December 18, 2008, responding to the Office Action mailed October 15, 2008 provided in the rejection of claims 1-11, 13-18, 20-26, and 28-33, wherein claims 1, 11, 18, and 26 have been amended.

Claims 1-11, 13-18, 20-26, and 28-33 remain pending in the application and which have been fully considered by the examiner.

Applicant's arguments with respect to claims currently amended have been fully considered but are moot in view of the new grounds of rejection – see *Chheda* et al. *and Chen* et al. - arts made of record, as applied hereto

### *Claim Rejections – 35 USC § 103(a)*

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

2. Claims 1-11, 13-14, 18, 20-21, 25-26, and 28-33 are rejected under 35 U.S.C. 103(a) as being unpatentable over Chheda et al. (Pub. No. US 2005/0114850 A1) (hereinafter 'Chheda' - art made of record) in view of Chen et al. *(Energy-Aware Compilation and Execution in Java-Enabled Mobile Devices, 2003 IEEE, pp. 1-8)* (hereinafter 'Chen' - art made of record)

3. **As to claim 1** (Currently Amended), Chheda discloses a method comprising:

- receiving a plurality of non-native instructions in a selected one of a source form (e.g., Fig. 1, element 10 – Compiler; [0068] – … a compiler 10 is a software system that programs circuitry to translate application form high-level programming language (e.g., C, C++, Java) into machine specific sequence of instructions …; Fig. 2, element 24 – Executable Re-Compiler; [0088] – … the executable re-compiler 30 can be integrated with various source-level compilers in the front-end 48 that have source files 46 as inputs …); and
- compiling the plurality of non-native instructions to generate object code for the non-native instructions, wherein compiling the plurality of non-native instructions includes replacing an object code segment from the generated object code with an alternative object code segment if the alternative object code segment improves at least a selected one of a power level required and an amount of energy required to execute the generated object code in a target execution environment (e.g., [0045] - … Many of these micro-operations are not necessary if there is related information extracted in the compiler that provides it, or if there is program information that enables a way to replace these micro-operations with more energy-efficient but equivalent ones; [0050] - … inserting a more

energy efficient instruction replacing an existing instruction and may include inserting control bits to control hardware structures)

Further, Chheda discloses analyzing and transforming a program executable at compile time such that a processor design objective is optimized (e.g., Abstract) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Energy-Aware Compilation and Execution in Java-Enabled Mobile Devices*, Chen discloses an intermediate form (e.g., Sec. 1 – Introduction, 5th Para - … The bytecodes are executed by Java Virtual Machine …)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Capra into the Chheda's system to further provide other limitations stated above in the Chheda system.

The motivation is that it would further enhance the Ecklund's system by taking, advancing and/or incorporating the Chen's system which offers significant advantages that our framework takes into account communication, computation and compilation energies to dynamically decide where to compile and execute a method (locally or remotely) and how to execute it (using interpretation or just-in-time compilation with different levels of optimizations) as once suggested by Chen (e.g., Abstract, 2nd Para)

4. **As to claim 2** (Original) (incorporating the rejection in claim 1), Chen discloses the method wherein said receiving comprises receiving the non-native

instructions in a byte code form (e.g., Sec. 1 – Introduction, 5th Para - … The bytecodes are executed by Java Virtual Machine …)

5. **As to claim 3** (Previously Presented) (incorporating the rejection in claim 1), Chheda discloses the method wherein said compiling comprises analyzing the object code segment for execution power level requirement, and determining whether an alternative object code segment with lower execution power level requirement is available (e.g., Fig. 3; [0075] – Disambiguating an Executable; [0076] - … the executable file is analyzed by a binary scanning tool in order to gain information about the various section and any other symbolic information that may be available. This information is used in order to create a version of the program based on energy-focused Binary Intermediate Format or BIF …; [0077]; [0078] – Once program analyses and optimizations have been run, the optimized BIF object can be converted back into an executable of the same type as the original one or possibly another type with a different instruction – see 42 in Fig. 3)

6. **As to claim 4** (Previously Presented) (incorporating the rejection in claim 1), Chheda discloses the method wherein said compiling comprises analyzing the object code segment for execution energy consumption, and determining whether an alternative object code segment with lower execution energy consumption is available (e.g., Fig. 3; [0075] – Disambiguating an Executable; [0076] - … the executable file is analyzed by a binary scanning tool in order to gain information about the various section and any other symbolic information that may be available. This information is used in order to create a version of the

program based on energy-focused Binary Intermediate Format or BIF …; [0077]; [0078] – Once program analyses and optimizations have been run, the optimized BIF object can be converted back into an executable of the same type as the original one or possibly another type with a different instruction – see 42 in Fig. 3)

7. **As to claim 5** (Previously Presented) (incorporating the rejection in claim 1), Chen discloses the method wherein the method further comprises

- executing the non-native instructions for an initial number of times using an interpreter (e.g., Sec. 1 – Introduction, 5th Para - … The bytecodes are executed by Java Virtual Machine (JVM). Simple JVMs execute Java applications by interpreting their bytecodes …; Sec. 3 – Analysis of Execution and Compilation Strategies, 2nd Para - … be identified by automatic tools that make use of profile information …); and
- performing said compiling only after executing the non-native instructions for said initial number of times (e.g., Sec. 1 – Introduction, 5th Para - … More sophisticated JVMs may selectively compile parts of the application's bytecodes into native code at runtime using a just-in-time (JIT) compiler)

8. **As to claim 6** (Previously Presented) (incorporating the rejection in claim 5), Chen discloses the method wherein the method further comprises determining the initial number of times the received non-native instructions are to be executed using the interpreter before performing compiling the received non-native instructions (e.g., Sec. 1 – Introduction, 5th Para - … The bytecodes are

executed by Java Virtual Machine (JVM). Simple JVMs execute Java applications by interpreting their bytecodes …; Sec. 3 – Analysis of Execution and Compilation Strategies, 2nd Para - … be identified by automatic tools that make use of profile information …)

9. **As to claim 7** (Previously Presented) (incorporating the rejection in claim 6), Chen discloses the method wherein the method further comprises

- monitoring said compiling for power level required to perform compilation;
- updating a current understanding of power level required for compilation (e.g., Sec. 1 – Introduction, 5th Para - … The bytecodes are executed by Java Virtual Machine (JVM). Simple JVMs execute Java applications by interpreting their bytecodes …; Sec. 3 – Analysis of Execution and Compilation Strategies, 2nd Para - … be identified by automatic tools that make use of profile information …; Sec. 2 – Target Platforms and Benchmarks, 3rd Para - … tracks the energy consumptions in the process core (data-path), on-chip caches …; Sec. 3.1 Analysis of Static Strategy; Fig. 6 – Energy consumption of three benchmarks with static execution strategies); and
- determining the initial number of times received non-native instructions are to be executed using the interpreter before compiling the received non-native instructions, if said monitoring observes a power level required for compilation to be different from the current understanding (e.g., Sec. 1 – Introduction, 5th Para - … The bytecodes are executed by Java Virtual

Machine (JVM). Simple JVMs execute Java applications by interpreting their bytecodes …; Sec. 3 – Analysis of Execution and Compilation Strategies, 2nd Para - … be identified by automatic tools that make use of profile information …)

10. **As to claim 8** (Previously Presented) (incorporating the rejection in claim 6), Chen discloses the method wherein the method further comprises

- monitoring said compiling for amount of energy required to perform an average compilation;
- updating a current understanding of amount of energy required for an average compilation (e.g., Sec. 1 – Introduction, 5th Para - … The bytecodes are executed by Java Virtual Machine (JVM). Simple JVMs execute Java applications by interpreting their bytecodes …; Sec. 3 – Analysis of Execution and Compilation Strategies, 2nd Para - … be identified by automatic tools that make use of profile information …; Sec. 2 – Target Platforms and Benchmarks, 3rd Para - … tracks the energy consumptions in the process core (data-path), on-chip caches …; Sec. 3.1 Analysis of Static Strategy; Fig. 6 – Energy consumption of three benchmarks with static execution strategies); and
- determining the initial number of times received non-native instructions are to be executed using the interpreter before compiling the received non-native instructions, if said monitoring observes an amount of energy required for compilation to be different from the current understanding

(e.g., Sec. 1 – Introduction, 5th Para - … The bytecodes are executed by Java Virtual Machine (JVM). Simple JVMs execute Java applications by interpreting their bytecodes …; Sec. 3 – Analysis of Execution and Compilation Strategies, 2nd Para - … be identified by automatic tools that make use of profile information …)

11. **As to claim 9** (Original) (incorporating the rejection in claim 1), Chen discloses the method wherein the generated object code comprises a plurality of native instructions, and the method further comprises

- monitoring execution of the generated object code for power level required to execute the native instructions; and
- updating power level requirements of selected ones of the native instructions if said monitoring observes power level requirements for the selected ones of the native instructions to be different from current understandings of the power level requirements of the selected ones of the native instructions (e.g., Sec. 1 – Introduction, 5th Para - … The bytecodes are executed by Java Virtual Machine (JVM). Simple JVMs execute Java applications by interpreting their bytecodes …; Sec. 3 – Analysis of Execution and Compilation Strategies, 2nd Para - … be identified by automatic tools that make use of profile information …; Sec. 2 – Target Platforms and Benchmarks, 3rd Para - … tracks the energy consumptions in the process core (data-path), on-chip caches …; Sec. 3.1

Analysis of Static Strategy; Fig. 6 – Energy consumption of three benchmarks with static execution strategies)

12. **As to claim 10** (Original) (incorporating the rejection in claim 1), Chen discloses the method wherein the generated object code comprises a plurality of native instructions, and the method further comprises

- monitoring execution of the generated object code for amount of energy required to execute the native instructions; and
- updating energy requirements of selected ones of the native instructions if said monitoring observes energy requirements for the selected ones of the native instructions to be different from current understandings of the energy requirements of the selected ones of the native instructions (e.g., Sec. 1 – Introduction, 5th Para - … The bytecodes are executed by Java Virtual Machine (JVM). Simple JVMs execute Java applications by interpreting their bytecodes …; Sec. 3 – Analysis of Execution and Compilation Strategies, 2nd Para - … be identified by automatic tools that make use of profile information …; Sec. 2 – Target Platforms and Benchmarks, 3rd Para - … tracks the energy consumptions in the process core (data-path), on-chip caches …; Sec. 3.1 Analysis of Static Strategy; Fig. 6 – Energy consumption of three benchmarks with static execution strategies)

13. **As to claim 11** (Currently Amended), Chheda discloses in an electronic device, a method of operation, comprising:

- receiving a plurality of non-native instructions (e.g., Fig. 1, element 10 – Compiler; [0068] – … a compiler 10 is a software system that programs circuitry to translate application form high-level programming language (e.g., C, C++, Java) into machine specific sequence of instructions …);
- compiling the non-native instructions into object code after executing the received non-native instructions for said initial number of times using the interpreter (e.g., [0045] - … Many of these micro-operations are not necessary if there is related information extracted in the compiler that provides it, or if there is program information that enables a way to replace these micro-operations with more energy-efficient but equivalent ones; [0050] - … inserting a more energy efficient instruction replacing an existing instruction and may include inserting control bits to control hardware structures)

Further, Chheda discloses analyzing and transforming a program executable at compile time such that a processor design objective is optimized (e.g., Abstract) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Energy-Aware Compilation and Execution in Java-Enabled Mobile Devices*, Chen discloses:

- determining an initial number of times to interpretively execute the non-native instructions based at least in part on one or more of an expected power level required to perform a compile or an expected energy required to perform the compile (e.g., Sec. 1 – Introduction, 5$^{th}$ Para - … The bytecodes are executed by Java Virtual Machine (JVM). Simple JVMs

execute Java applications by interpreting their bytecodes …; Sec. 3 – Analysis of Execution and Compilation Strategies, 2nd Para - … be identified by automatic tools that make use of profile information …; Sec. 2 – Target Platforms and Benchmarks, 3rd Para - … tracks the energy consumptions in the process core (data-path), on-chip caches …; Sec. 3.1 Analysis of Static Strategy; Fig. 6 – Energy consumption of three benchmarks with static execution strategies);

- executing the non-native instructions for the initial number of times using an interpreter (e.g., Sec. 1 – Introduction, 5th Para - … The bytecodes are executed by Java Virtual Machine (JVM). Simple JVMs execute Java applications by interpreting their bytecodes …; Sec. 3 – Analysis of Execution and Compilation Strategies, 2nd Para - … be identified by automatic tools that make use of profile information …);

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Capra into the Chheda's system to further provide other limitations stated above in the Chheda system.

The motivation is that it would further enhance the Ecklund's system by taking, advancing and/or incorporating the Chen's system which offers significant advantages that our framework takes into account communication, computation and compilation energies to dynamically decide where to compile and execute a method (locally or remotely) and how to execute it (using interpretation or just-in-

time compilation with different levels of optimizations) as once suggested by Chen (e.g., Abstract, 2nd Para)

14. **As to claim 13** (Previously Presented) (incorporating the rejection in claim 11), Chen discloses the method wherein the method further comprises

- monitoring said compiling for a compilation requirement employed in determining the initial number of times the received non-native instructions are to be executed using the interpreter before compiling (e.g., Sec. 1 – Introduction, 5th Para - … The bytecodes are executed by Java Virtual Machine (JVM). Simple JVMs execute Java applications by interpreting their bytecodes …; Sec. 3 – Analysis of Execution and Compilation Strategies, 2nd Para - … be identified by automatic tools that make use of profile information …); and
- updating a current understanding of the compilation requirement if said monitoring observes the compilation requirement to be different from the current understanding (e.g., Sec. 1 – Introduction, 5th Para - … The bytecodes are executed by Java Virtual Machine (JVM). Simple JVMs execute Java applications by interpreting their bytecodes …; Sec. 3 – Analysis of Execution and Compilation Strategies, 2nd Para - … be identified by automatic tools that make use of profile information …; Sec. 2 – Target Platforms and Benchmarks, 3rd Para - … tracks the energy consumptions in the process core (data-path), on-chip caches …; Sec. 3.1

Analysis of Static Strategy; Fig. 6 – Energy consumption of three benchmarks with static execution strategies)

15. **As to claim 14** (Original) (incorporating the rejection in claim 11), Chen discloses the method wherein the generated object code comprises a plurality of native instructions, and the method further comprises

- monitoring execution of the generated object code for execution requirements of the native instructions; and
- updating execution requirements of selected ones of the native instructions if said monitoring observes execution requirements for the selected ones of the native instructions to be different from current understandings of the execution requirements of the selected ones of the native instructions (e.g., Sec. 1 – Introduction, 5th Para - … The bytecodes are executed by Java Virtual Machine (JVM). Simple JVMs execute Java applications by interpreting their bytecodes …; Sec. 3 – Analysis of Execution and Compilation Strategies, 2nd Para - … be identified by automatic tools that make use of profile information …; Sec. 2 – Target Platforms and Benchmarks, 3rd Para - … tracks the energy consumptions in the process core (data-path), on-chip caches …; Sec. 3.1 Analysis of Static Strategy; Fig. 6 – Energy consumption of three benchmarks with static execution strategies)

16. **As to claim 18** (Currently Amended), Chheda discloses an article of manufacture comprising:

- a computer readable medium; and
- a plurality of instructions designed to implement a runtime manager equipped to receive a plurality of non-native instructions (e.g., Fig. 1, element 10 – Compiler; [0068] – … a compiler 10 is a software system that programs circuitry to translate application form high-level programming language (e.g., C, C++, Java) into machine specific sequence of instructions …; Fig. 2, element 24 – Executable Re-Compiler; [0088] – … the executable re-compiler 30 can be integrated with various source-level compilers in the front-end 48 that have source files 46 as inputs …),
- execute the non-native instructions for an said initial number of times using an interpreter, and invoke a compiler to compile the non-native instructions into object code after executing the received non-native instructions for said initial number of times using the interpreter (e.g., Fig. 2, element 24 – Executable Re-Compiler; [0088] – … the executable re-compiler 30 can be integrated with various source-level compilers in the front-end 48 that have source files 46 as inputs …; [0045] - … Many of these micro-operations are not necessary if there is related information extracted in the compiler that provides it, or if there is program information that enables a way to replace these micro-operations with more energy-efficient but equivalent ones; [0050] - … inserting a more energy efficient

instruction replacing an existing instruction and may include inserting control bits to control hardware structures)

Further, Chheda discloses analyzing and transforming a program executable at compile time such that a processor design objective is optimized (e.g., Abstract) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Energy-Aware Compilation and Execution in Java-Enabled Mobile Devices*, Chen discloses:

- determine an initial number of times to interpretively execute the non-native instructions based at least in part on one or more of an expected power level required to perform an average compile or an expected energy required to perform the average compile (e.g., Sec. 1 – Introduction, 5th Para - … The bytecodes are executed by Java Virtual Machine (JVM). Simple JVMs execute Java applications by interpreting their bytecodes …; Sec. 3 – Analysis of Execution and Compilation Strategies, 2nd Para - … be identified by automatic tools that make use of profile information …)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Capra into the Chheda's system to further provide other limitations stated above in the Chheda system.

The motivation is that it would further enhance the Ecklund's system by taking, advancing and/or incorporating the Chen's system which offers significant advantages that our framework takes into account communication, computation

and compilation energies to dynamically decide where to compile and execute a method (locally or remotely) and how to execute it (using interpretation or just-in-time compilation with different levels of optimizations) as once suggested by Chen (e.g., Abstract, 2nd Para)

17. **As to claim 20** (Original) (incorporating the rejection in claim 18), Chheda discloses the article wherein the runtime manager is further equipped to

- monitor said compiling for a compilation requirement employed in determining the initial number of times received non-native instructions are to be executed before compiling; and
- update a current understanding of the compilation requirement if said monitoring observes the compilation requirement to be different from the current understanding (e.g., Sec. 1 – Introduction, 5th Para - … The bytecodes are executed by Java Virtual Machine (JVM). Simple JVMs execute Java applications by interpreting their bytecodes …; Sec. 3 – Analysis of Execution and Compilation Strategies, 2nd Para - … be identified by automatic tools that make use of profile information …; Sec. 2 – Target Platforms and Benchmarks, 3rd Para - … tracks the energy consumptions in the process core (data-path), on-chip caches …; Sec. 3.1 Analysis of Static Strategy; Fig. 6 – Energy consumption of three benchmarks with static execution strategies)

18. **As to claim 21** (Original) (incorporating the rejection in claim 18), Chen discloses the article wherein the generated object code comprises a plurality of native instructions, and the runtime manager is further equipped to

- monitor execution of the generated object code for execution requirements of the native instructions; and
- update execution requirements of selected ones of the native instructions if said monitoring observes execution requirements for the selected ones of the native instructions to be different from current understandings of the execution requirements of the selected ones of the native instructions (e.g., Sec. 1 – Introduction, 5th Para - … The bytecodes are executed by Java Virtual Machine (JVM). Simple JVMs execute Java applications by interpreting their bytecodes …; Sec. 3 – Analysis of Execution and Compilation Strategies, 2nd Para - … be identified by automatic tools that make use of profile information …; Sec. 2 – Target Platforms and Benchmarks, 3rd Para - … tracks the energy consumptions in the process core (data-path), on-chip caches …; Sec. 3.1 Analysis of Static Strategy; Fig. 6 – Energy consumption of three benchmarks with static execution strategies)

19. **As to claim 25** (Original) (incorporating the rejection in claim 22), Chen discloses the system wherein the apparatus further comprises a wireless communication interface to receive the non-native instructions (e.g., Sec. 1 Introduction, 4th - … to optimize both computation and communication energy …)

20. **As to claim 26** (Currently Amended), Chheda discloses a system, comprising:

- a communication interface to receive a plurality of non-native instructions;

a storage medium coupled to the communication interface, and having stored therein a plurality of instructions designed to implement a runtime manager equipped to determine an initial number of times to

- invoke a compiler to compile the non-native instructions into object code after executing the received non-native instructions for said initial number of times using the interpreter (e.g., Fig. 2, element 24 – Executable Re-Compiler; [0088] – … the executable re-compiler 30 can be integrated with various source-level compilers in the front-end 48 that have source files 46 as inputs …; [0045] - … Many of these micro-operations are not necessary if there is related information extracted in the compiler that provides it, or if there is program information that enables a way to replace these micro-operations with more energy-efficient but equivalent ones; [0050] - … inserting a more energy efficient instruction replacing an existing instruction and may include inserting control bits to control hardware structures); and

- a processor coupled to the storage medium to execute the instructions implementing the runtime manager (e.g., [0020])

Further, Chheda discloses analyzing and transforming a program executable at compile time such that a processor design objective is optimized (e.g., Abstract) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Energy-Aware Compilation and Execution in Java-Enabled Mobile Devices*, Chen discloses:

- interpretively execute the non-native instructions based at least in part on one or more of an expected power level required to perform an average compile or an expected energy required to perform the average compile, execute the received non-native instructions for the initial number of times using an interpreter (e.g., Sec. 1 – Introduction, 5th Para - … The bytecodes are executed by Java Virtual Machine (JVM). Simple JVMs execute Java applications by interpreting their bytecodes …; Sec. 3 – Analysis of Execution and Compilation Strategies, 2nd Para - … be identified by automatic tools that make use of profile information …)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Capra into the Chheda's system to further provide other limitations stated above in the Chheda system.

The motivation is that it would further enhance the Ecklund's system by taking, advancing and/or incorporating the Chen's system which offers significant advantages that our framework takes into account communication, computation and compilation energies to dynamically decide where to compile and execute a

method (locally or remotely) and how to execute it (using interpretation or just-in-time compilation with different levels of optimizations) as once suggested by Chen (e.g., Abstract, 2nd Para)

21. **As to claim 28** (Previously Presented) (incorporating the rejection in claim 26), Chen discloses the system wherein the runtime manager is further equipped to

- monitor said compiling for a compilation requirement employed in determining the initial number of times received non-native instructions are to be executed using the interpreter before compiling; and
- update a current understanding of the compilation requirement if said monitoring observes the compilation requirement to be different from the current understanding (e.g., Sec. 1 – Introduction, 5th Para - … The bytecodes are executed by Java Virtual Machine (JVM). Simple JVMs execute Java applications by interpreting their bytecodes …; Sec. 3 – Analysis of Execution and Compilation Strategies, 2nd Para - … be identified by automatic tools that make use of profile information …; Sec. 2 – Target Platforms and Benchmarks, 3rd Para - … tracks the energy consumptions in the process core (data-path), on-chip caches …; Sec. 3.1 Analysis of Static Strategy; Fig. 6 – Energy consumption of three benchmarks with static execution strategies)

22. **As to claim 29** (Original) (incorporating the rejection in claim 26), Chen discloses the system wherein the generated object code comprises a plurality of native instructions, and the runtime manager is further equipped to

- monitor execution of the generated object code for execution requirements of the native instructions; and
- update execution requirements of selected ones of the native instructions if said monitoring observes execution requirements for the selected ones of the native instructions to be different from current understandings of the execution requirements of the selected ones of the native instructions (e.g., Sec. 1 – Introduction, 5th Para - … The bytecodes are executed by Java Virtual Machine (JVM). Simple JVMs execute Java applications by interpreting their bytecodes …; Sec. 3 – Analysis of Execution and Compilation Strategies, 2nd Para - … be identified by automatic tools that make use of profile information …; Sec. 2 – Target Platforms and Benchmarks, 3rd Para - … tracks the energy consumptions in the process core (data-path), on-chip caches …; Sec. 3.1 Analysis of Static Strategy; Fig. 6 – Energy consumption of three benchmarks with static execution strategies)

23. **As to claim 30** (Original) (incorporating the rejection in claim 26), Chen discloses the system wherein the communication interface is a wireless communication interface (e.g., Sec. 1 – Introduction, 4th Para – In a mobile

wireless device, it is important to optimize both computation and communication energy ...)

24. **As to claim 31** (Previously Presented) (incorporating the rejection in claim 11), Chen discloses the method wherein the method further comprises determining an initial number of times to interpretively execute the non-native instructions based at least in part on a size of the non-native instructions (e.g., Sec. 1 – Introduction, 5th Para - … The bytecodes are executed by Java Virtual Machine (JVM). Simple JVMs execute Java applications by interpreting their bytecodes …; Sec. 3 – Analysis of Execution and Compilation Strategies, 2nd Para - … be identified by automatic tools that make use of profile information …)

25. **As to claim 32** (Previously Presented) (incorporating the rejection in claim 18), please refer to claim **31** above, accordingly.

26. **As to claim 33** (Previously Presented) (incorporating the rejection in claim 26), please refer to claim **31** above, accordingly.

### *Claim Rejections – 35 USC § 102(e)*

The following is quotation of 35 U.S.C. 102(e) which form the basis for all obviousness rejections set forth in this office action:

> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the

international application designated the United States and was published under Article 21(2) of such treaty in the English language.

27. Claims 15-17 and 22-24 are rejected under 35 U.S.C. 102(e) as being anticipated by Chheda.

28. **As to claim 15** (Previously Presented), Chheda discloses an article of manufacture comprising:

- a computer readable medium; and
- a plurality of instructions designed to implement a compiler to compile non-native instructions to generate object code for the non-native instructions, and replace an object code segment with an alternative object code segment if the alternative object code segment improves at least a selected one of a power level required and an energy required to execute the generated object code (e.g., Fig. 2, element 24 – Executable Re-Compiler; [0088] – … the executable re-compiler 30 can be integrated with various source-level compilers in the front-end 48 that have source files 46 as inputs …; [0045] - … Many of these micro-operations are not necessary if there is related information extracted in the compiler that provides it, or if there is program information that enables a way to replace these micro-operations with more energy-efficient but equivalent ones; [0050] - … inserting a more energy efficient instruction replacing an existing instruction and may include inserting control bits to control hardware structures)

29. **As to claim 16** (Previously Presented) (incorporating the rejection in claim 15), please refer to claim **3** as set forth accordingly.

30. **As to claim 17** (Previously Presented) (incorporating the rejection in claim 15), please refer to claim **4** as set forth accordingly.

31. **As to claim 22** (Previously Presented), Chheda discloses a system, comprising:

a storage medium having stored therein a plurality of instructions implementing a compiler to

- compile non-native instructions to generate object code for the non-native instructions (e.g., Fig. 2, element 24 – Executable Re-Compiler; [0088] – … the executable re-compiler 30 can be integrated with various source-level compilers in the front-end 48 that have source files 46 as inputs …), and
- replace an object code segment with an alternative object code segment if the alternative object code segment improves at least a selected one of a power level required and an energy required to execute the generated object code (e.g., [0045] - … Many of these micro-operations are not necessary if there is related information extracted in the compiler that provides it, or if there is program information that enables a way to replace these micro-operations

with more energy-efficient but equivalent ones; [0050] - … inserting a more energy efficient instruction replacing an existing instruction and may include inserting control bits to control hardware structures); and

- a processor coupled to the storage medium to execute the instructions implementing the compiler (e.g., [0020])

32. **As to claim 23** (Previously Presented) (incorporating the rejection in claim 22), Chheda discloses the system wherein said compiler analyzes the object code segment for execution power level requirement, and determining whether an alternative object code segment with lower execution power level requirement is available (e.g., Fig. 3; [0075] – Disambiguating an Executable; [0076] - … the executable file is analyzed by a binary scanning tool in order to gain information about the various section and any other symbolic information that may be available. This information is used in order to create a version of the program based on energy-focused Binary Intermediate Format or BIF …; [0077]; [0078] – Once program analyses and optimizations have been run, the optimized BIF object can be converted back into an executable of the same type as the original one or possibly another type with a different instruction – see 42 in Fig. 3)

33. **As to claim 24** (Previously Presented) (incorporating the rejection in claim 22), Chheda discloses the system wherein said compiler analyzes the object code segment for execution energy consumption, and determining whether an

alternative object code segment with lower execution energy consumption is available (e.g., Fig. 3; [0075] – Disambiguating an Executable; [0076] - … the executable file is analyzed by a binary scanning tool in order to gain information about the various section and any other symbolic information that may be available. This information is used in order to create a version of the program based on energy-focused Binary Intermediate Format or BIF …; [0077]; [0078] – Once program analyses and optimizations have been run, the optimized BIF object can be converted back into an executable of the same type as the original one or possibly another type with a different instruction – see 42 in Fig. 3)

***Conclusion***

34. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Ben C Wang/
Ben C. Wang
Examiner, Art Unit 2192

/Tuan Q. Dam/
Supervisory Patent Examiner, Art Unit 2192